Scientia Iranica D (2012) 19 (3), 690–695

Sharif University of Technology

Scientia Iranica

Transactions D: Computer Science & Engineering and Electrical Engineering

www.sciencedirect.com

Application of Karnaugh map for easy generation of error correcting codes

M. Tabandeh

School of Electrical Engineering, Sharif University of Technology, Tehran, P.O. Box 11155-4363, Iran

Received 7 September 2009; revised 19 June 2010; accepted 22 October 2010

KEYWORDS

Karnaugh map;
Linear codes;
Hamming distance;
Hamming type code;
Code word;
S-map;
K-code.

Abstract With increasing use of data transmission between digital systems as well as subsystems, the need for more reliable communication is felt. In this research, a new approach to the linear error correcting codes is introduced. Among advantages of this technique are simpler code construction and also decoding algorithm, as well as easier understanding of the basic concept. Based on an earlier paper and also recent work done, we first discuss briefly the use of Karnaugh map and its advantages in constructing simple codes. We then prove a theorem on application of the map to multiple error correcting codes. Using these results, we propose a simple technique that leads to obtaining a code with more capabilities. We also discuss another advantage of using the Karnaugh map in error correcting codes. As an example, we discuss a special case of generating a double error correcting code and using Karnaugh map features to give it extended capabilities.

1. Introduction

Simplified schemes for generation of error correcting codes, using Karnaugh map, has been previously proposed [1]. Also, analytical work done by Muller [2] and Reed [3] has had significant contributions to the field. Reviriego et al. have done an interesting research in soft error detection and correction [4]. Payandeh et al. have considerable contributions to secure channel coding [5]. In [6], Biberstein and Etzion have achieved a profound theoretical research on single error correction and double-adjacent-error detection. Helgert and Stinaff have approached the problem of correcting codes by using tables of minimum distance between them [7]. Also, deep theoretical work has been done on the subject and books published [8–10]. However, simpler handling of the Karnaugh map made us choose this technique that by visualizing the error positions in the map, offers remarkable ease in code generation and also understanding. In this paper, our goal is to do a deeper study of the relationship of error coding with the Karnaugh map and obtain simple and useful results in generating error correcting codes and possibly add extra features. As a result of the Gray code arrangement in the rows and columns, we can use visual reasoning rather than having to go through mathematical concepts and sometimes complex definitions and proofs leading to the same results. An n variable Karnaugh map is subdivided into 2^n (n is the number of variables handled by the map) squares. Two adjacent squares in the Karnaugh map differ in their Boolean representations by a change in only one variable, and therefore their binary representations will have a Hamming distance of one. This feature is exploited in our coding scheme to find out the right codes assigned to pertinent correctable errors.

To obtain a better understanding of the use of Karnaugh map in producing error correcting codes, we first use an example based on the technique explained in [1]. Let us consider a (15, 11) Hamming code that corrects one error. Figure 1 displays the parity test matrix for this code. This code contains a total number of 15 bits consisting of 11 data bits denoted as x_i ($i = 1, 2, \ldots, 11$), and four parity bits denoted as p_j ($j = 1, 2, 3, 4$). Let us consider now four subsets, S_k ($k = 1, 2, 3, 4$), of the set of all the data and parity bits in the code defined as follows (Figure 2):

1. Every subset S_k contains two or more data bits x_i. Also, one parity bit p_k is associated to each subset S_k (Figure 2).
2. The value of the exclusive parity bit for each subset is determined so that the sum of all members of that subset will equal zero (case of even parity).

E-mail address: tabandeh@sharif.edu.
Peer review under responsibility of Sharif University of Technology.

Production and hosting by Elsevier

1026-3098 © 2012 Sharif University of Technology. Production and hosting by Elsevier B.V. Open access under CC BY-NC-ND license.
doi:10.1016/j.scient.2011.08.030

$$H=\begin{array}{c} \begin{matrix} p_1 & p_2 & p_3 & p_4 & x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & x_8 & x_9 & x_{10} & x_{11} \end{matrix} \\ \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix} \end{array} \begin{matrix} 0 \\ 1 \\ 0 \\ 1 \end{matrix} \begin{matrix} : S_1 \\ : S_2 \\ : S_3 \\ : S_4 \end{matrix}$$

Figure 1: The parity test matrix of our code.

$$S_1=\{p_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8\}$$
$$S_2=\{p_2, x_5, x_6, x_7, x_8, x_9, x_{10}, x_{11}\}$$
$$S_3=\{p_3, x_1, x_2, x_3, x_6, x_7, x_9, x_{10}\}$$
$$S_4=\{p_4, x_1, x_3, x_4, x_7, x_8, x_{10}, x_{11}\}$$

Figure 2: Subsets S_j and their members.

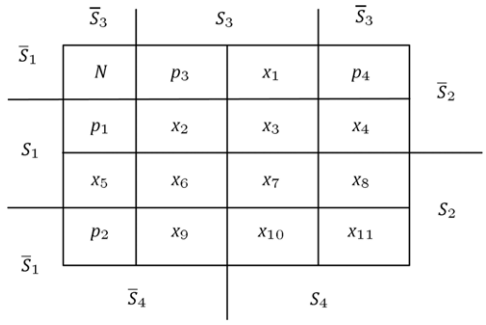

Figure 3: Karnaugh map display of subsets S_j and their members.

We can therefore display membership of bits in different subsets in a Karnaugh map. An example of this for the above code is shown in Figure 3. Assume now that after a transmission, one bit, say x_6, is received erroneously, i.e. its value has changed. We therefore will note at the receiver that the parity check for S_1, S_2 and S_3 will not satisfy the second condition above. In other words, for these sets, the sum (XOR) of the value of all members will be equal to 1. Therefore, we will have $C_4C_3C_2C_1 = 0111$ where every C_m $(m = 1, 2, 3, 4)$ is the parity check bit for the corresponding S_m. One can easily note from the Karnaugh map of Figure 3 that the erroneous bit specified by 0111 is the square denoted by and assigned to x_6. In Figure 3, the square with the letter N represents a correct code received.

In the Karnaugh maps used for generating codes correcting multiple errors, we should assign one square to every single bit in the codeword as well as to every combination of bit errors we wish to correct by this code. As an example, for a double error correcting code, the combinations of any two bits in the code such as x_ix_j, x_ip_j and p_ip_j must be assigned exclusive squares in the relevant map. It was proved that, in order to obtain a Karnaugh map correcting two errors, we need to assign relevant x_i in the S_k map with a distance of three. This is easily obtained by a Hamming type code correcting single errors [1].

In this paper, we want to treat some special cases using the Karnaugh map. As an example, suppose we want to transmit three data bits with the possibility of correcting two errors. To go forward, we need to define new terms. In Section 2, we present the necessary definitions needed for the rest of the paper.

2. Definitions

To avoid confusion for the reader, we are going to define some terms that we will use frequently throughout this paper.

Code word. A sequence of $(r + k)$ bits comprising k data (information) and r parity bits $(k = 1, 2, \ldots, n, r = 1, 2, \ldots, m)$ transmitted together with a predefined arrangement over a transmission channel.

Subset. We consider the total number of information bits in a codeword as the *set* of data bits, S. We then define different *subsets* S_i of the above set as overlapping, each containing at least two data bits (x_j, x_k) and one parity bit (p_i) exclusive to itself.

S-map. A Karnaugh map intended for assigning data bits to its squares and whose variables consist of *parity bits of* a code under consideration.

K-Square. A K-square is any of the subdividing squares in the S-map. A square assigned to bit x_i of a code word will be denoted as y_i, and the square assigned to p_j will be referred to as q_j.

K-code. The set of bits (code) specifying one square in the S-map will be called the K-code for that square (e.g. $S_1S_2S_3S_4S_5S_6 = 100011$). A K-code represents membership of the corresponding data bit x_i or parity bit p_j to different subsets (for example, membership of a data bit to S_1, S_5 and S_6 in the above example). We may also specify a data bit x_i by its S-map representation y_i or equivalently by its binary representation (or K-code) Y_i. Also, a parity bit p_j will be specified by its S-map square q_j or its K-code Q_j.

We can now state the following notes as direct results of linear algebra and error correcting codes.

Note 1. Every K-square q_j has exactly one non zero element in its K-code (Q_j) representation. Also, every square y_i must have at least two non zero elements in its K-code (Y_i) representation meaning the membership of x_i to at least two subsets.

Note 2. Let the K-codes Y_i and Q_j denote the corresponding squares y_i and q_j in the S-map. Then, the squares corresponding to the combination of y_i and q_j will be obtained by summing (EX-ORing) the relevant Y_i and Q_j.

In the next section, we will prove a useful theorem about multiple errors that will help us obtain a limit on the maximum number of data bits (given a number of parity bits) in a code. Also, an example will clarify certain features of our technique.

3. Multiple error correction

In a previous paper, we noted that by determining y_i's in the S-map, we obtain K-codes for the corresponding data bits in a code. Now we want to look into the conditions to be satisfied by a certain code if it is to correct e errors. We proposed a systematic approach to produce an $(e + 1)$ error correcting code using direct results of a code correcting e errors [1]. The following theorem generalizes this problem.

Theorem. *Let C be a code correcting e errors. For each i; let Y_i be the corresponding K-code of the data bit x_i (y_i position in the S-map for C). Then, the binary sum of m different K-codes Y_i has a Hamming weight greater or equal to $2e - m + 1$.*

Proof. To prove this theorem, we assume the K-code (Y_i) or its equivalent y_i square corresponding to the data bit x_i in the codeword. We must have separate squares assigned to each single error or every error combination possible. Therefore, we must make sure that the squares assigned to every combination of k bits $(k = 1, 2, \ldots, e)$ have not been previously assigned to

any other error in one bit or combination of bits in the code. In other words, no overlap should exist between them. Therefore, we should have:

$$\sum_{m=1}^{e} Y_m \neq \sum_{n=1}^{e-k} Y_n + \sum_{l=1}^{k} Q_l, \tag{1}$$

where Σ denotes modulo 2 summing (EXORing) of K-codes.

Relation (1) must be valid for all values of k. Note also that for $k = 0$ {where the last term of Relation (1) is considered non existent}, we can state that, the sum of $2eY$'s should contain at least one non zero element. Similarly, for $k = 1$, we can state that the sum of $(2e - 1)Y$'s should contain at least two 1's and so forth. For the general case, one can deduce that the sum of $(2e - m + 1)Y$'s should have a Hamming weight of m, and vice versa, and the sum of mY's must have a hamming weight of at least $(2e - m + 1)$. □

As an example of using the results obtained in the above theorem, let us assume that we have 3 bits of information and we want to add necessary redundancy bits to them in order to obtain a code correcting all the double errors. One can easily compute that we will need seven parity bits [11–13].

In a code using seven parity bits, we know that we can employ four information bits with the desired specifications [1]. We can therefore predict that with only three information bits, we would lose partially the capabilities of the code. That is, interpreted by a number of unused squares in the S-map that could possibly be used to give extra capabilities to the code. For example, the code could probably detect or even correct some of three bit errors. To explore this, let us first investigate the possibility of correcting all the three bit errors. The total number of squares in the Karnaugh map needed for this code should be at least equal to:

$$\sum_{i=0}^{3} \binom{10}{i} = 176.$$

Note that this number is greater than 2^7 and therefore it is mathematically impossible to have such a code. Now consider a code with the capability of correcting only a subset of all three bit errors on 10 bit code words, say some particular three error cases. For example, let us consider burst error on three consecutive bits of the code that has a larger probability of occurrence than other three bit errors and may possibly take place as the result of a momentary channel malfunction or instant power failure. To produce such a code, we need first to determine an arrangement of data and parity bits in the sequence of oncoming code bits and then proceed to single out every three consecutive bits to assign a square in the S-map. To study the possibility of such a code, let us first examine the simplest way of sequencing bits in this code, that is separating totally parity and information bits in the code in such a manner to enable us to determine maximum number of three consecutive (parity) bit errors prior to assigning squares to information bits. We therefore would have a code word of the form:

$$\mathbf{x_1x_2x_3}p_1p_2p_3p_4p_5p_6p_7.$$

We now consider a seven bit S-map. In this map, we first determine all the squares assigned to one and two parity bit errors. Then, we specify the squares corresponding to those special burst error cases we wish to correct. For example, $x_1x_2x_3$, $x_2x_3p_1$ etc. are three bit burst error candidates that we propose to correct in our code. We specify each of these three bit errors by a single letter as follows:

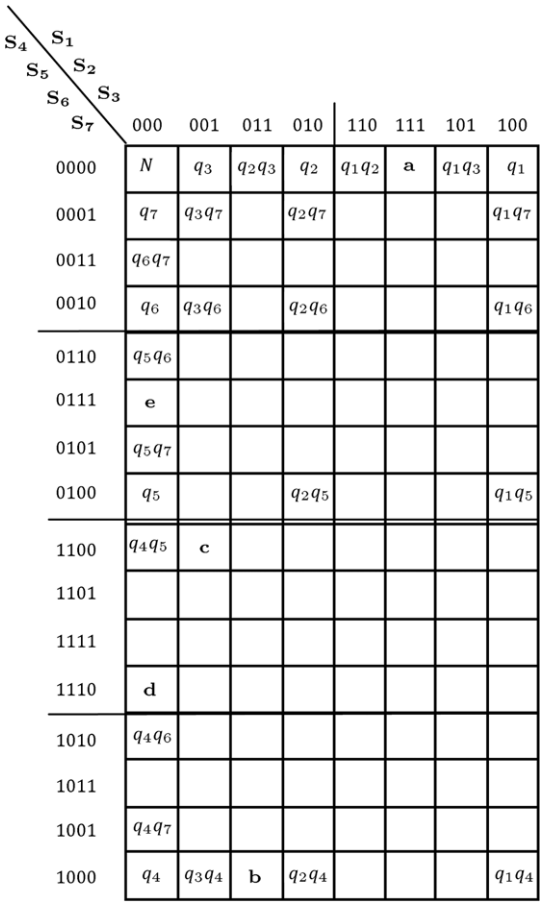

Figure 4: Karnaugh map with relevant parity bit error positions. All one- and two- bit positions and also special three bits are indicated.

$$\mathbf{a} = x_1x_2x_3, \quad \mathbf{b} = x_2x_3p_1, \quad \mathbf{c} = x_3p_1p_2,$$
$$\mathbf{d} = p_1p_2p_3, \quad \mathbf{e} = p_2p_3p_4, \quad \mathbf{f} = p_3p_4p_5,$$
$$\mathbf{g} = p_4p_5p_6, \quad \mathbf{h} = p_5p_6p_7.$$

Figure 4 displays the relevant S-map with squares assigned to single, double and also three consecutive parity bits, as well as their combinations of two and also the special three bit errors desired to be corrected in this design.

At this stage, we can start assigning the information bits (y_i's). As we know, in order that a code be capable of correcting two errors, the corresponding K-codes for its data bits must have a Hamming distance of at least three. Therefore, a systematic way of obtaining such a K-code would be to generate all the sixteen code words of a Hamming type seven bit code words and then use them as K-codes. This code can be obtained using a three variable eight square S-map as explained in [1]. The code words for this code are noted as D_j in the relevant squares of the S-map of Figure 5. Any one of these code words is therefore a candidate K-code for a double error correcting code. The main challenge here is that we want to add other features to the code in order to extend its error correction capabilities. Note also that in Figure 5, any square D_j can be obtained as the sum of two other squares D_m and D_n.

Now we want to pick up the right D_j's from Figure 5. The theorem proved above suggests that every y_i must be a member of at least four subsets (i.e. its K-code must contain at least four non zero elements). Moreover, giving the capability of correcting some three bit errors requires that Y_i's be placed

$S_4S_5S_6S_7$ \ $S_1S_2S_3$	000	001	011	010	110	111	101	100
0000	$\mathbf{D_1}$	q_3	q_2q_3	q_2	q_1q_2	$\mathbf{D_2}$	q_1q_3	q_1
0001	q_7	q_3q_7		q_2q_7				q_1q_7
0011	q_6q_7	$\mathbf{D_3}$			$\mathbf{D_4}$			
0010	q_6	q_3q_6		q_2q_6				q_1q_6
0110	q_5q_6		$\mathbf{D_5}$					$\mathbf{D_6}$
0111	**e**							
0101	q_5q_7			$\mathbf{D_7}$			$\mathbf{D_8}$	
0100	q_5			q_2q_5				q_1q_5
1100	q_4q_5	$\mathbf{D_9}$			$\mathbf{D_{10}}$			
1101								
1111	$\mathbf{D_{11}}$					$\mathbf{D_{12}}$		
1110	**d**							
1010	q_4q_6			$\mathbf{D_{13}}$			$\mathbf{D_{14}}$	
1011								
1001	q_4q_7		$\mathbf{D_{15}}$					$\mathbf{D_{16}}$
1000	q_4	q_3q_4	**b**	q_2q_4				q_1q_4

Figure 5: A Karnaugh map with the squares distant of three indicated by D_i. Note that squares indicated by **a** and **c** have been replaced by $\mathbf{D}_2$ and $\mathbf{D}_9$.

$S_4S_5S_6S_7$ \ $S_1S_2S_3$	000	001	011	010	110	111	101	100
0000	N	q_3	q_2q_3	q_2	q_1q_2	**c**	q_1q_3	q_1
0001	q_7	q_3q_7		q_2q_7				q_1q_7
0011	q_6q_7	y_1y_2					**a**	
0010	q_6	q_3q_6	y_3q_5	q_2q_6				q_1q_6
0110	q_5q_6	y_3q_2	y_3	y_3q_3		y_3q_1		
0111	**g**		y_3q_7			y_2q_4		
0101	q_5q_7		**h**					
0100	q_5	q_3q_5	y_3q_6	q_2q_5	y_1q_4			q_1q_5
1100	q_4q_5	**e**		y_1q_1	y_1	y_1q_3		y_1q_2
1101					y_1q_7	y_2q_6		
1111		**b**	y_2q_1		y_2q_3	y_2	y_2q_2	
1110	**f**		y_3q_4		y_1q_6	y_2q_7		
1010	q_4q_6						y_1y_3	
1011						y_2q_5		
1001	q_4q_7							y_2y_3
1000	q_4	q_3q_4	**d**	q_2q_4	y_1q_5			q_1q_4

Figure 6: Karnaugh map for the final code sequence with no overlap of three bit errors.

in squares with a distance of three from the three parity bit locations. This would mean that another three bit error including y_i under consideration would not coincide with that special error. Nor a double error should coincide or even be adjacent to any of those three error squares we have singled out, since that square must remain exclusive. Rephrasing the above statements, we can set forth a set of conditions that K-codes corresponding to x_i ($i = 1, 2, 3$) must satisfy. These conditions are:

1. Y_i's must all have at least four non zero elements (this limits our choice of Y_i's to D_4, D_5, D_8, D_{10}, D_{11}, D_{12}, D_{14} and D_{15}).
2. In the binary representation form of Y_i's alone or their sum, we should not have three adjacent 1's, unless they have six or more non zero elements. This means that the corresponding y_i in the S-map would occupy a square adjacent to a three consecutive parity square reserved to correct a special error and cause that square not to be exclusive.
3. The sum of all three Y_i's should have at least four 1's as it is a K-code for three *data* bit error and should not be confused with a three consecutive *parity* bit error (that would have three adjacent 1's in its K-code).

Using the candidate squares indicated in Figure 5 and selecting three squares among them whose K-code has at least four non zero elements, say D_5, D_{10} and D_{12}, we obtain an acceptable solution only for a regular double error correcting code. However, this code will not be able to correct three adjacent errors as we planned, since it does not satisfy all the conditions above (the square corresponding to the three bit error $y_1y_2y_3$ would fall adjacent to the double error q_2q_5 and therefore this square will not correspond to a unique three bit error of $y_1y_2y_3$ but also to that of $q_2q_5q_7$). However, moving one of data bits apart from the others solves this problem. We therefore modify the sequence to obtain:

$$\mathbf{x_1x_2}p_1p_2p_3p_4p_5p_6p_7\mathbf{x_3}, \tag{2}$$

that will satisfy the conditions and therefore yield an acceptable solution. Figure 6 displays the new map relevant to the code sequence of Relation (2). Note that this map satisfies the conditions above and therefore corrects all three adjacent bits as well. The letters used in the S-map of Figure 6 correspond to three consecutive bit errors as follows:

$$\mathbf{a} = y_1y_2q_1, \quad \mathbf{b} = y_2q_1q_2, \quad \mathbf{c} = q_1q_2q_3,$$
$$\mathbf{d} = q_2q_3q_4, \quad \mathbf{e} = q_3q_4q_5, \quad \mathbf{f} = q_4q_5q_6,$$
$$\mathbf{g} = q_5q_6q_7, \quad \mathbf{h} = q_6q_7y_3.$$

In general, finding a solution for a similar problem requires considering the conditions ruling the case and designing the relevant algorithm using the results of the theorem proved in Section 3 to narrow down the choices.

In Section 4, we introduce a theory that will help us find the minimum number of errors correctable by a designed code.

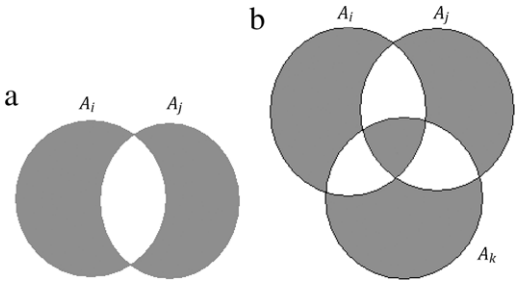

Figure 7: Graphical interpretation of (a) A_{ij}, and (b) A_{ijk}.

4. The number of parity bits required of the code

Consider a code word consisting of groups of x and p bits, namely x_i and p_j. As we saw in the first section, subsets S_i of code bits are defined by their members. We know that each data bit (x_i) is a member of at least two subsets, while parity bits (p_j) belong each to only one subset.

Before proceeding to compute a lower bound on the number of parity bits, we need to make another definition as follows:

Definition. Corresponding to each data bit x_k, we define A_k as the set of all the subsets S_j containing x_k.

As an example, consider the S-map of Figure 2. For x_6 we can then write:

$$A_6 = \{S_1, S_2, S_3\}.$$

Now, according to the theorem proved in Section 3, if Y_i has at least $2e$ non zero elements in its K-code, then we can state that x_i is a member of at least $2e$ subsets. We express this mathematically in the following way.

$$m(A_i) \geq 2e,$$

where $m(A_i)$ denotes the number of members of A_i. On the other hand, we know that the logic sum $(Y_i + Y_j)$ must have at least $(2e - 1)$ non zero elements. This would mean that there are at least $(2e - 1)$ subsets (A_{ij}) in such a way that only one of x_i or x_j would be a member of it, that is:

$$m(A_{ij}) \geq 2e - 1; \quad A_{ij} \equiv (A_i \cup Aj) - (A_i \cap A_j).$$

The shaded areas in Figure 7(a) show graphical interpretation of A_{ij}. In this figure, A_i is the set of the subsets containing x_i and likewise, A_j is the set of all the subsets containing x_j. In a similar way, for the sum of $(Y_i + Y_j + Y_k)$, we have:

$$n(A_{ijk}) \geq 2e - 2; \quad A_{ijk} \equiv (A_{ij} \cup A_k) - (A_{ij} \cap A_k),$$

and the graphical interpretation of A_{ijk} is shown in Figure 7(b).

As an example of application of the above theory, we list the inequalities obtained for the case of $k = 3$. With reference to Figure 8, we have:

$$n(A_1) \geq 2e \Rightarrow n_1 + n_{12} + n_{13} + n_{123} \geq 2e, \quad (3)$$
$$n(A_2) \geq 2e \Rightarrow n_2 + n_{12} + n_{23} + n_{123} \geq 2e, \quad (4)$$
$$n(A_3) \geq 2e \Rightarrow n_3 + n_{13} + n_{23} + n_{123} \geq 2e, \quad (5)$$
$$n(A_{12}) \geq 2e - 1 \Rightarrow n_1 + n_2 + n_{13} + n_{23} \geq 2e - 1, \quad (6)$$
$$n(A_{23}) \geq 2e - 1 \Rightarrow n_2 + n_3 + n_{13} + n_{12} \geq 2e - 1, \quad (7)$$
$$n(A_{13}) \geq 2e - 1 \Rightarrow n_3 + n_1 + n_{12} + n_{23} \geq 2e - 1, \quad (8)$$
$$n(A_{123}) \geq 2e - 2 \Rightarrow n_1 + n_2 + n_3 + n_{123} \geq 2e - 2. \quad (9)$$

In the above relations, n_1 is the number of subsets containing only x_1 and n_{m1} is the number of subsets containing both x_m and x_1 and so forth. Also, the + sign implies regular summing and not the logical Exclusive Or (XOR) function of the values. We can therefore write:

$$r = n_1 + n_2 + n_3 + n_{12} + n_{13} + n_{23} + n_{123},$$

and since we wish to optimize this code, our goal will be to minimize r.

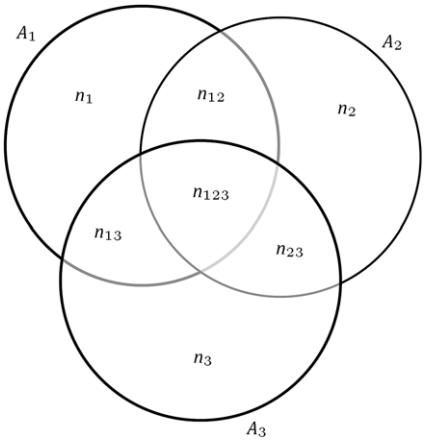

Figure 8: Graphical representation for Relations (3) through (9).

The above inequalities for $k = 3$ and various values of e yield the following results.

e	2	3	4	5
r	7	10	14	17

As an example, for $(k, e, r) = (3, 2, 7)$, we have:

$$n_1 = n_2 = n_3 = n_{12} = n_{13} = n_{23} = n_{123} = 1.$$

One possible solution for relevant K-codes is:

$$Y_1 = 1001101, \quad Y_2 = 0101011, \quad Y_3 = 0010111.$$

Also, for the case of $k = 4$, we obtain the following results

e	2	3	4	5
r	7	10	15	18

And for the case of $(k, e, r) = (4, 2, 7)$, a possible solution for relevant K-codes could be:

$$Y_1 = 1011001, \quad Y_2 = 0100111,$$
$$Y_3 = 1110100, \quad Y_4 = 1101010.$$

One can note that in order to obtain the actual code words for the above case, we will have to add seven parity bits to the original four data bits. As a matter of fact, as long as the transmitter and receiver are using the same protocol about data and parity bit placement in the code, the relevant codeword could have its bits in any desired order.

As the number of data bits, k, increases, the number of inequalities above will increase as well. Also, obtaining an optimum value for r becomes more time consuming. In fact, solving these equations requires some type of linear programs with r as the cost function to be minimized. The reason the calculations were discussed here, however was only to prove the correctness of the results obtained. In general, we will not have to go through these involved calculations.

5. Construction of codes

For a code system, once we have obtained relevant K-codes for a code system, we should determine different redundancy bits according to the rules of subset membership in order to

$$G = \begin{bmatrix} 1101001101 \\ 1011011000 \\ 0001101011 \end{bmatrix}$$

Figure 9: The generating matrix for the designed code.

produce the actual code. As an example, consider the double error correcting code with the special three errors defined in section two and the relevant S-map of Figure 6. For the membership of information bits in different subsets in the map of Figure 6, we can write

$A_1 = \{S_1, S_2, S_4, S_5\}$,
$A_2 = \{S_1, S_2, S_3, S_4, S_5, S_6, S_7\}$,
$A_3 = \{S_2, S_3, S_5, S_6\}$.

The code words based on bit arrangement shown in Relation (2) will therefore be generated according to the following relation:

$$\bar{C} = \bar{W} \times \bar{G},$$

where $\bar{W} = \{x_1, x_2, x_3\}$ is the vector representing information bits, $\bar{G}$ is the generating matrix shown in Figure 9 and $\bar{C}$ represents the set of all the code words generated by our code system.

A general algorithm for producing code words should follow the rules below:

1. Define all the subsets S_i of the code system specified by their data bit members and their exclusive parity bits.
2. Compute the value of the parity bit for each subset.
3. Arrange the code words with the relevant bits in their correct position.

In our particular case, the eight-code words produced according to the above rules are:

	$\mathbf{x_1}$	$\mathbf{x_2}$	p_1	p_2	p_3	p_4	p_5	p_6	p_7	$\mathbf{x_3}$
$C_1 =$	**0**	**0**	0	0	0	0	0	0	0	**0**
$C_2 =$	**0**	**0**	0	1	1	0	1	1	0	**1**
$C_3 =$	**0**	**1**	1	0	1	1	0	1	1	**0**
$C_4 =$	**0**	**1**	1	0	0	1	0	0	1	**1**
$C_5 =$	**1**	**0**	1	1	0	1	1	0	0	**0**
$C_6 =$	**1**	**0**	1	0	1	1	0	1	0	**1**
$C_7 =$	**1**	**1**	0	0	1	0	0	1	1	**0**
$C_8 =$	**1**	**1**	0	1	0	0	1	0	1	**1**

6. Decoding algorithm

The code words produced by our technique consist of separated data and parity bits with a special arrangement. A decoding algorithm should therefore follow the simple rules as below:

1. Form different subsets S_i by subdividing the received code words accordingly.
2. Compute the check bits C_i for each subset S_i. A check bit C_i is obtained by taking the parity of all members of the subset S_i.
3. Find the vector $C_nC_{n-1}C_{n-2}\ldots C_2C_1$ of all check bits.
4. Consider the vector above as a K-code and along with the definition of error positions in the S-map, find the exact error(s) location in the code word. Then correct the erroneous bits specified.

7. Conclusion

Although in some cases, nonlinear codes may be more powerful and flexible in correcting errors, simplicity and higher speed inherent to linear codes are far more advantageous for certain applications. This was our main motivation in pursuing this research.

In this paper, using an earlier idea of the Karnaugh map application to error correcting codes, we first proved a theorem about the conditions K-codes of the data bits must satisfy in an e error correcting code and then, using the results of this theorem, we produced a special code correcting all two errors and added extra features; correcting some particular three bit errors using the new technique. The results obtained for the similar cases match exactly those analytically obtained [1]. However, here, we have added more capabilities to our code. We noted also that by solving the linear inequalities, we could compute lower bounds on the number of redundancy bits for optimal linear error correcting codes. Of course, the computation of limits on the number of parity bits becomes more and more complex and time consuming as the number of data bits or errors to be corrected increase. The reason we discussed this matter in this paper was only to prove the correctness of our procedure. In practice, we will not have to do any of these calculations. An example was discussed thoroughly and relevant code generated.

References

[1] Ward, R. and Tabandeh, M. "Error correction and detection, a geometric approach", *The Computer Journal*, 27(3), pp. 246–253 (1984).
[2] Muller, D.E. "Application of Boolean algebra to switching circuit design and to error detection", *IRE Transactions on Electronic Computers*, EC-3, pp. 6–12 (1954).
[3] Reed, I.S. "A class of multiple-error correcting codes and the decoding scheme", *IRE Transactions on Electronic Computers*, EC-3, pp. 6–12 (1954).
[4] Reviriego, P., Maestro, J.A., O'Donnell, A. and Bleakley, C.J. "Soft error detection and correction for FFT based convolution using different block lengths", *15th IEEE International on-Line Testing Symposium*, pp. 139–143 (June 2009).
[5] Payandeh, A., Ahmadian, M. and Aref, M.R. "An adaptive secure channel coding scheme for data transmission over LEO satellite channels", *Scientia Iranica*, 13(4), pp. 373–378 (2006).
[6] Biberstein, M. and Etzion, T. "Optimal codes for single-error correction, double—adjacent-error detection", *IEEE Transactions on Information Theory*, pp. 2188–2191 (2000).
[7] Helgert, H.J. and Stinaff, R.D. "Minimum-distance bounds for binary linear codes", *IEEE Transactions on Information Theory*, IT-19, pp. 344–356 (1973).
[8] Morelos-Zaragoza, R.H., *The Art of Error Correcting Coding*, John Wiley, West Sussex (2002).
[9] Lin, S. and Costello Jr., D.J., *Error Control Coding*, 2nd ed., Prentice Hall, Englewood Cliffs (2004).
[10] Klove, T., *Codes for Error Detection*, World Scientific, Singapore (2007).
[11] Blahut, R.E., *Theory and Practice of Error Control Code*, Adison-Wesley MA2 (1983).
[12] Macwilliams, F.J. and Sloane, N.J.A., *The Theory of Error Correcting Codes*, North-Holland, Amsterdam (1993).
[13] Jia, H. "What makes NP-complete problems hard"?, Ph.D. Thesis, University of New Mexico, Albuquerque (2007).

Mahmoud Tabandeh received his electronic engineering degree from Institut National des Sciences Appliquées (INSA) de Lyon (France), his M.S. degree from Louisiana State University (LSU) and his Ph.D. degree from the university of California, Berkeley (UCB).

He is currently an associate Professor in the School of Electrical Engineering, Sharif University of Technology, Tehran, Iran. His research interests include digital systems, hardware and software in general and image processing in particular.